Optimization of complex robot applications under thermal constraints

Matthieu Guilbert, Pierre-Brice Wieber, Luc Joly

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Optimization of complex robot applications under thermal constraints

Matthieu Guilbert — Pierre-Brice Wieber — Luc Joly

N° 6074

December 2006

Thème NUM

ISSN 0249-6399

INRIA
RHÔNE-ALPES

Optimization of complex robot applications under thermal constraints

Matthieu Guilbert*, Pierre-Brice Wieber †, Luc Joly ‡

Thème NUM — Systèmes numériques
Projets Bipop et Stäubli Robotics Faverges

Rapport de recherche n° 6074 — December 2006 — 27 pages

Abstract: This paper deals with minimum time trajectory optimization along a specified path subject to thermal constraints. We point out here that robots are often integrated in complex robotic cells, and the interactions between the robot and its environment are often difficult or even impossible to modelize. The structure of the optimization problem allows us to decompose the optimization in two levels, the first one being based on models and results of the theory of the calculus of variations, the second one being based on measurements and derivative free algorithms. This decomposition allows us to optimize the velocity profiles efficiently without knowing in advance the interactions between the robot and is environment. We propose here two numerical algorithms for these two levels of the decomposition which show good convergence properties. The resulting optimal velocity profiles are 5 to 10% faster than classical ones, and have been executed on successfully on a real Stäubli Rx90 manipulator robot.

Key-words: Robotics, Trajectory, numerical optimization, calculus of variations, thermal model, derivative free optimization, augmented Lagrangian

* Stäubli Robotics Faverges
† INRIA Rhône Alpes
‡ Stäubli Robotics Faverges

Optimisation d'applications robotiques complexes avec contraintes de température

Résumé : Ce document se concentre sur l'optimisation de trajectoire le long d'une trajectoire prédéfinie tout en prenant en compte les limitations thermiques des actionneurs des robots. On souligne ici le fait que le robot est souvent intégré dans une cellule robotique et que les interactions entre le robot et son environment sont souvent difficiles voire impossibles à modéliser. La structure du problème d'optimisation nous permet de décomposer l'optimisation en deux niveaux, le premier étant basé sur des modèles et sur des résultats de la théorie du calcul de variations, le second étant basé sur des mesures et des algorithmes d´optimisation sans dérivée. Cette décomposition nous permet d'optimiser au mieux les profils tout en connaissant mal les interactions entre le robot et son environement. Ce travail nous a conduit à écrire un algorithme numérique qui a de bonnes propriétés de convergence, le trajectoires optimales ont été exécutées avec succès sur un robot Stäubli Rx90. Les simulations numériques et les expériences nous ont permis de comparer différentes techniques d'optimisation, on a alors pu conclure que la méthode que nous proposons donne de bons résultats comparée aux méthodes plus classiques.

Mots-clés : Robotique, trajectoire, optimisation numérique, calculs de variations, modèle thermique, optimisation sans dérivées, Lagrangien augmenté

Contents

1 Introduction

The programming of industrial robots is generally based on the operator's experience, regardless of the system's precise dynamics and relevant optimization criteria. And due to the complexity of robots and manufacturing systems, even highly qualified operators can only reach a limited level of efficiency. A better exploitation of the performances of robots integrated in manufacturing systems can only be achieved therefore by using computer aided optimization methods. Now, previous results on trajectory optimization usually focus on generating trajectories with minimum time or minimum energy criteria subject to the actuators' limitations without taking into account all the unmodelized interactions between the robot and its environment usually composed of several machine tools, pallets, etc... Moreover, these results usually consider limitations such as maximum velocities, accelerations or torques [Bes92] [LLO91] [Hol84] [BDG85] [LCL83] which don't reflect all the real limitations of a robot such as overheating, wearing and breaking. We propose here to derive an algorithm for optimizing velocity profiles in order to obtain a minimum cycle time while taking into account thermal constraints on one side, and all the unmodelized interactions between the robot and its environment, on the other side.

We will first derive a temperature model in section 2, then we will show in section 3 that the special structure of the proposed optimization problem allows decomposing it in two levels. We will develop in section 4 an optimal profile generator which corresponds to the first level using some calculus of variations, and we will develop in section 5 derivative free optimization method for dealing with the unmodelized interactions between the robot and its environment which corresponds to the second level. We will finally test these algorithms in section 6 with numerical simulations and experiments on a real industrial robot.

2 Temperature prediction for robotic systems

Minimizing the duration of robotic applications usually induces strong demands on the mechanical and electrical parts of the robots. Wearing and overheating are some of the classical consequences of these demands, and we will focus here on the increase of temperature. Since a high temperature can cause damages, this increase of temperature must be controlled and since the rise of temperature is a slow phenomenon (it can take more than 5 hours to stabilize), sensors can't be used to measure in advance the future stabilized temperature, reason why we need a thermal model to predict it. Since most robotic applications are cyclic, it is possible to derive a model which predicts the stabilized temperature corresponding to a given cycle once this cycle is known precisely enough. To predict this temperature, the references [Ltd95], [Ltd88] and [cL89] propose to take into account the loss by Joule effect in the motors. Only the reference [Cor01] takes into account both the loss of the motors and the loss in the mechanical parts of the actuators. Heat transfers between gears, motors, and other parts of the robot can be described by conduction, convection and radiation phenomena [TP98], but in practice, these three transfer modes are simultaneous and not easy to separate: their study is therefore often empirical.

A thermal model will be derived in section 2.1 which only takes into account the conduction phenomenon (this is the major heat transfer in our system), then the validity of the model will be tested in section 2.2.

2.1 Thermal model of the system

In order to predict the temperature of the robot, we will predict in fact the temperature at different points considered to be representative of the system from a thermal point of view. Moreover the articulations in an industrial robot are often enclosed in casings: there exist therefore strong thermal coupling between actuators. Specifically, we will identify our model on a Stäubli Rx90 in which the actuators are enclosed by pairs. Six heat sources can be distinguished then, 3 by actuator:

$$\begin{pmatrix} \Delta T_1 \\ \Delta T_2 \end{pmatrix} = A \begin{pmatrix} I_1 \\ I_2 \end{pmatrix} + B \begin{pmatrix} V_1 \\ V_2 \end{pmatrix} + \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} \tag{1}$$

with ΔT_j the elevation of temperature of the representative point j, A and B two constant matrices which represent the thermal resistances in the different materials, γ_1 and γ_2 two constant vectors representing the constant loss of the coils in the brakes, I_1 and I_2 representing the loss by Joule effect of the coils depending on the articular torque (since the current is globally proportional to the torques), V_1 and V_2 representing the loss due to friction in the gears depending on the articular velocity (since the motor velocity is globally proportional to the articular velocity), with:

$$I_j = \frac{1}{t_f} \int_0^{t_f} \Gamma_j(t)^2 dt, \text{ and } V_j = \frac{1}{t_f} \int_0^{t_f} \dot{q}_j(t)^2 dt$$

where $\Gamma_j(t)$ is the articular torque and $\dot{q}_j(t)$ is the articular velocity of the j^{th} axis of the robot.

The initial definition of this model has been based on physical considerations on Joule effects, friction, conduction, dissipation and other thermic effects, but a precise modelling of all these effects on a system as complex as a manipulator robot can be out of reach, and maybe not even useful for our purpose. This is why we restrict ourselves to the model (1) which can be considered to already reflect correctly the global behaviour of the true system, as shown in the next section.

2.2 Identification and validation of the model

To identify the constants in this model for a given robot, 100 different trajectories have been executed on this robot with different current and velocity mean values (I_j and V_j). For each trajectory, the stabilized temperature is measured after 6 hours of execution. The parameters can be identified then with a least squares procedure. The reliability of the model can be evaluated by studying the error prediction of this model and by calculating the confidence intervals of the identified parameters.

The general tendency of the prediction error can be seen in figure 1 to show non linear behaviors for low currents and velocities. This obviously means that not all the physical phenomena have been modeled in equation (1). However despite the empirical design of this model, it gives good predictions with a mean error of 5%.

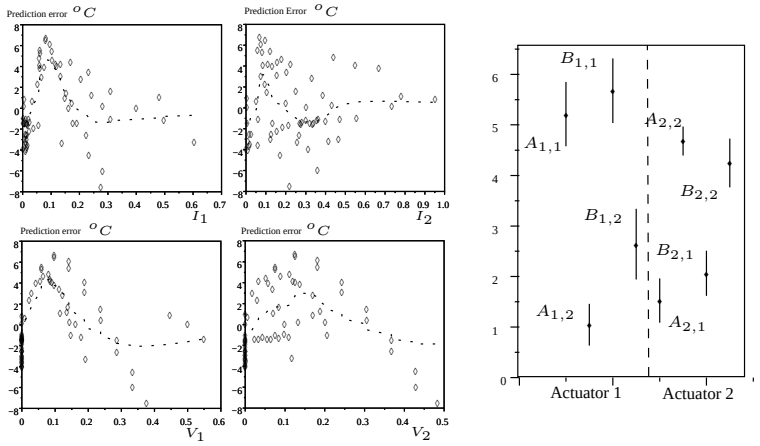

Figure 1: Left: Prediction error according to I_1, I_2, V_1, V_2. Right: Confidence intervals of the elements of the matrix A and B

Confidence intervals [WW90] of the thermal resistances appearing in the matrices A and B have also been calculated. Statistically speaking, we are 95% sure that the parameters are in the intervals represented in figure 1. Since the intervals don't cross the zero axis, all the identified parameters appear to have an influence on the predicted temperature, so all of them need to be present in equation (1).

3 Optimization of complex robotic applications

3.1 General structure of complex robotic applications

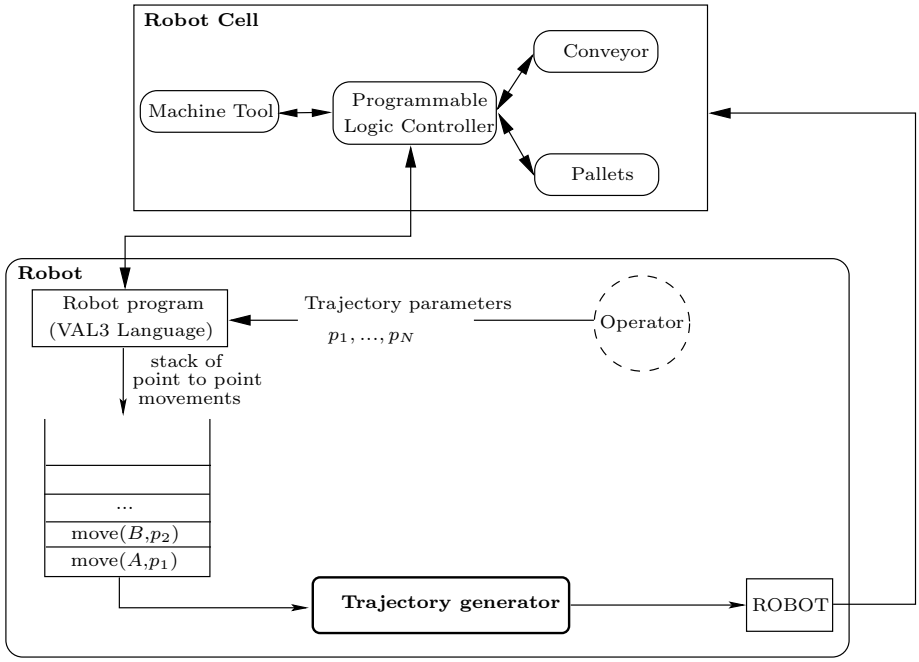

Figure 2: Classical structure of a complex robotic application

We are interested here in minimizing the cycle time of a complex robotic application without exceeding a maximum authorized temperature of the robot:

$$\min\ (t_{N+1} - t_1) \tag{2}$$

$$\frac{1}{t_{N+1} - t_1} \int_{t_1}^{t_N+1} (A\,\Gamma^2 + B\,\dot{q}^2)\,dt + \gamma + T_{amb} \leq T_{max}, \tag{3}$$

with T_{amb} the ambient temperature, T_{max} the maximum authorized temperature, t_1 the start time of the cycle and t_{N+1} the end time of the cycle.

But in order to predict the temperature through the constraint (3) according to the thermal model (1), we are supposed to have a perfect knowledge of both the robot and the complete task it needs to realize. Unfortunately this task usually involves several machine tools, conveyors and pallets, all scheduled by a complex Programmable Logic Controller (PLC) which form together a complete robot cell that can be hard or even impossible to modelize. The interactions between the robot and the robot cell imply especially to be able to react to events which aren't always perfectly known in advance: small differences in the timings of the machine tools, small differences in the pick and place motions.

Reacting to such events not perfectly known in advance is usually done [BM03] by decomposing the trajectory in a series of point to point motions which are prepared only shortly ahead of time in order to allow quick reactions, and put in a stack as shown in Figure 2. Doing so, the trajectory generator needs therefore to work online with only a limited view of the task to be realized, what doesn't seem to be compatible at first sight with constraints such as the constraint (3) which needs to be taken into account over the whole cycle.

In order to be able to optimize somehow such complex and not perfectly known robotic applications, we will make the key assumption that they are "globally cyclic". More precisely, we will suppose that the differences between the cycles of these robotic tasks are small enough with respect to the optimal problem (2)-(3) so that some knowledge can be gathered cycle after cycle about the task, and used successfully in optimizing it.

3.2 Decomposability of the optimization problem

Let's make now an observation on the decomposability of the optimization problem (2)-(3) that is going to be of importance for solving it sucessfully in complex robotic applications such as the one described in Figure 2. If we decompose a cyclic movement of the robot in N distinct motions $\mathcal{M}_i$ over time intervals $[t_i, t_{i+1}]$, we can consider $t_{i+1} - t_i = d(\mathcal{M}_i)$, $\int_{t_i}^{t_{i+1}} (A\Gamma^2 + B\dot{q}^2)dt = T(\mathcal{M}_i)$, so that the problem (2)-(3) can be written as:

$$\begin{cases} \min\limits_{(\mathcal{M}_1,\dots,\mathcal{M}_N)} \sum\limits_{i=1}^{N} d(\mathcal{M}_i) \\ \sum\limits_{i=1}^{N} T(\mathcal{M}_i) - (T_{max} - \gamma - T_{amb})d(\mathcal{M}_i) \leq 0. \end{cases} \tag{4}$$

Following the ressource decomposition method generally used for large scale problems [BGLS03], this optimization problem appears to be decomposable, what means that it is equivalent to:

$$\begin{cases} \min\limits_{(p_1,\dots,p_N)} \sum\limits_{i=1}^{N} d_i^*(p_i) \\ \sum\limits_{i=1}^{N} p_i \leq 0, \end{cases} \tag{5}$$

with:

$$d_i^*(p_i) = \begin{cases} \min\limits_{\mathcal{M}_i} d(\mathcal{M}_i) \\ T(\mathcal{M}_i) - (T_{max} - \gamma - T_{amb})d(\mathcal{M}_i) \leq p_i. \end{cases} \tag{6}$$

Here, the optimization problems (6) correspond to optimizing each motion $\mathcal{M}_i$ independently from the others except for the ressource p_i which is allocated globally by the optimization problem (5). And these ressources p_i correspond to an energy increase allowed for each interval $[t_i, t_{i+1}]$.

3.3 Two levels of optimization

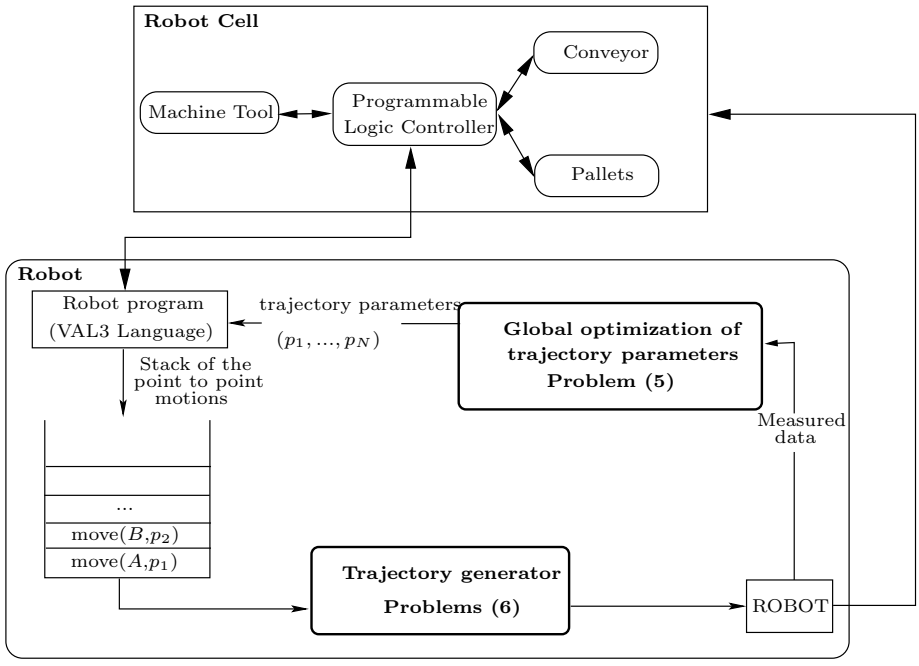

Figure 3: Two levels of optimization

(i) Initialize the parameters $(p_1, ..., p_N)$
(ii) Execute a cycle of the robotic application
(ii-a) Solve the problems (6) to generate each point to point motion
(ii-b) Execute these motions and measure t_c, $\dot{q}$ and Γ
(iii) Find a new set of parameters $(p_1, ..., p_N)$ through the optimization problem (7)
(iv) go to step (ii) until an optimal set of parameters is found

Table 1: General guidelines for the two levels of optimization

A nice property of the decomposition (5)-(6) of the optimization problem (2)-(3) is that it fits perfectly the structure of the robotic application shown in Figure 2. Indeed, we can observe in Figure 3 that the optimization problems (6) can be made to correspond to the trajectory generation with only a limited view of the task, while the global knowledge of this task is dealt with by the optimization problem (5). The key point in doing so is that different optimization methodologies can be used on these two distinct levels of optimization: the optimization problems (6) work on point to point motions which are sufficiently known in advance so that efficient model based optimization methods can be used, as will be shown in section 4, whereas the interaction with the imperfectly known robot cell can be treated entirely at the level of problem (5), where specific methods for dealing with imprecise problems, such as derivative free optimization can be used, as will be shown in section 5.

This way, dynamic models of the robot can be used to evaluate the duration $d(.)$ and the temperature increase $T(.)$ corresponding to a movement $\mathcal{M}_i$ when solving th optimization problems (6). But in order to take into account the unmodelized parts of the whole robotic cell when solving the ressource allocation problem (5), these quantities will need to be evaluated with the help of measures gathered directly on the robot. The cycle time can be measured directly, but not the stabilized temperature, as discussed earlier in section 2, so this temperature will still need to be estimated with the help of the model (1), but based on real measures of the torques and speeds of the actuators of the robot. Doing so, the problem (5) turns into:

$$\begin{cases} \min\limits_{(p_1,...,p_N)} t_c \\ \int_0^{t_c} (A\Gamma^2 + B\dot{q}^2)\, dt - (T_{max} - \gamma - T_{amb})\, t_c \leq 0, \end{cases} \tag{7}$$

where t_c, Γ and $\dot{q}$ are the cycle time, the torque and the velocity, all measured directly on the robot. Such an optimization of complex robotic applications with measured data would follow therefore the guidelines of Table 1.

4 An optimal profile generator

The description of the general optimization algorithm in section 3 has shown that the trajectory generator needs to solve a succession of local problems (6) which deal with point to point motions. Since we aren't interested here in optimizing the geometric path of the trajectory (for industrial reasons such as security), we will focus in this section on the optimization of point to point motions along a specified geometric path. We will propose first of all the calculation of an analytical solution in a simple case in section 4.1, then a generic spline based algorithm in section 4.2 to compute a numerical approximation of the solution in the general case. In this section, we will only consider a point to point motion beginning at time $t = 0$ and finishing at time $t = t_f$ without loss of generality.

4.1 Analytical solution in a simple case

4.1.1 Definition of the problem

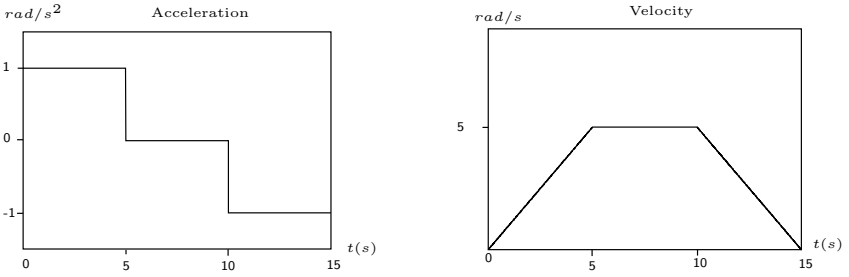

Figure 4: Acceleration and velocity for a BANG-zero-BANG optimal trajectory

Minimum time control problems in robotics are classicaly solved with the help of BANG-BANG or BANG-zero-BANG solutions [BH75]. Such optimal solutions appear when bounds are expressed on the control variables. Figure 4 shows such an optimal solution when bounds are expressed on the acceleration and the velocity. They can be derived with the help of the Maximum Principle of Pontryagin [BH75], and they present jumps of the control variables from one bound to another, what explains their name. In our case, the bounds are not directly expressed on the control variables but on the temperature, therefore such profiles aren't correct answers to our problem. To find an analytical solution when the bounds are expressed on the temperature, we consider a simple movement of a horizontal axis of our robot. By consequence, the system's dynamics presents no gravity effects, a constant inertia, no centrifugal and coriolis forces, what leads to the simple dynamic model:

$$\Gamma = J\ddot{q} + F_v\dot{q} + F_s \tag{8}$$

where Γ is the articular torque, $\ddot{q}$ the acceleration, $\dot{q}$ the velocity, J the inertia of the whole system and F_v and F_s the viscous and Coulomb friction (a constant here since we will consider a trajectory where the sign of the velocity doesn't change). In terms of function to minimize and constraints, the problem we need to solve here is:

$$\min\ t_f = \int_0^{t_f} 1\,dt \tag{9}$$

subject to:

$$\frac{1}{t_f}\int_0^{t_f} a(J\ddot{q} + F_v\dot{q} + F_s)^2 + b\,\dot{q}^2\,dt = T_{max} - c \tag{10}$$

$$\int_0^{t_f} \dot{q}\,dt = q_f - q_0 \tag{11}$$

with q_0 and q_f the initial and final position of the axis, a, b, c the constants of the thermal model (always strictly positive) and T_{max} the maximal authorized temperature. Note that the constraint (10) is an equality instead of an inequality since we consider that it will be an active constraint for this trajectory.

This is a minimization problem subject to isoperimetric constraints in which the end point isn't fixed [Pin93]. Lagrange multipliers λ_1 and λ_2 need to be introduced and we want to find the saddle points of:

$$\int_0^{t_f} \left(1 + \frac{\lambda_1}{t_f}(a(J\ddot{q} + F_v\dot{q} + F_s)^2 + b\dot{q}^2) + \lambda_2\dot{q}\right) dt = \int_0^{t_f} F(t, \dot{q}, \ddot{q})dt \tag{12}$$

The necessary condition in this case is the Euler-Lagrange differential equation:

$$\frac{\partial F(t, \dot{q}, \ddot{q})}{\partial \dot{q}} - \frac{d}{dt}\left(\frac{\partial F(t, \dot{q}, \ddot{q})}{\partial \ddot{q}}\right) = 0 \tag{13}$$

with boundary conditions:

$$\begin{cases} \dot{q}(0) = 0, \\ \dot{q}(t_f) = 0. \end{cases} \tag{14}$$

Since t_f isn't fixed, the following transversality condition must hold:

$$F(0) - \ddot{q}(0)\frac{\partial F}{\partial \ddot{q}}(0) = 0. \tag{15}$$

4.1.2 Calculation of the optimal velocity profile

The first step to calculate the optimal velocity profile is to solve the Euler-Lagrange equation. Since

$$\frac{\partial F}{\partial \dot{q}} = 2\lambda_1\frac{aF_v^2 + b}{t_f}\dot{q} + \frac{2\lambda_1 aJF_v}{t_f}\ddot{q} + \frac{2a\lambda_1 F_s F_v}{t_f} + \lambda_2, \tag{16}$$

$$\frac{\partial F}{\partial \ddot{q}} = \frac{2\lambda_1 aJ^2}{t_f}\ddot{q} + \frac{2\lambda_1 aJF_v}{t_f}\dot{q} + \frac{2a\lambda_1 F_s J}{t_f}, \tag{17}$$

$$\frac{d}{dt}\left(\frac{\partial F}{\partial \ddot{q}}\right) = \frac{2\lambda_1 aJ^2}{t_f}\dddot{q} + \frac{2\lambda_1 aJF_v}{t_f}\ddot{q}, \tag{18}$$

the Euler-Lagrange equation (13) becomes

$$\frac{-2\lambda_1 aJ^2}{t_f}\dddot{q} + \frac{2\lambda_1(aF_v^2 + b)}{t_f}\dot{q} + \frac{2a\lambda_1 F_s F_v}{t_f} + \lambda_2 = 0 \tag{19}$$

or in another way

$$\dddot{q} - r\dot{q} = C(\lambda_1, \lambda_2) \tag{20}$$

with $r = \dfrac{aF_v^2 + b}{aJ^2}$ and $C(\lambda_1, \lambda_2) = \dfrac{F_s F_v}{J^2} + \dfrac{\lambda_2}{2a\lambda_1 J^2}$.

Note that if $\lambda_1 = 0$, the Euler-Lagrange equation (19) gives $\lambda_2 = 0$ and the problem (12) degenerates: we can fairly consider therefore that λ_1 is different from zero. Integrating the equation (20) leads to solutions of the form:

$$\dot{q}(t) = \zeta\,\sinh(\sqrt{r}\,t) + \nu\,\cosh(\sqrt{r}\,t) - \frac{C(\lambda_1, \lambda_2)}{r}. \tag{21}$$

Now, the constants ζ, ν, λ_1, λ_2 and t_f can be determined with the boundary conditions (14), the constraints (10)-(11) and the transversality condition (15). In fact, we don't need to calculate the Lagrange multipliers λ_1 and λ_2 explicitely, and determining ζ, ν and C is enough to define the optimal solution:

$$\begin{cases} \zeta(t_f) = (q_f - q_i)\dfrac{\sqrt{r}(\cosh(\sqrt{r}\,t_f) - 1)}{-2\cosh(\sqrt{r}\,t_f) + 2 + t_f\sqrt{r}\sinh(\sqrt{r}\,t_f)}, \\ \nu(t_f) = -(q_f - q_i)\dfrac{\sqrt{r}\sinh(\sqrt{r}\,t_f)}{-2\cosh(\sqrt{r}\,t_f) + 2 + t_f\sqrt{r}\sinh(\sqrt{r}\,t_f)}, \\ C(t_f) = r\nu. \end{cases} \tag{22}$$

Note that these constants are defined with respect to the final time t_f which is computed by solving numerically the constraint (10).

The figure 5 shows such an optimal velocity profile on a Stäubli Rx90 for a movement of its first axis from -2.26 rad to $+2.26$ rad. After solving the equation (10), we find $t_f = 1.38s$. Since the jerk appears in the necessary condition (20), the acceleration is continuous and differentiable everywhere, what helps to avoid vibrations. Note that the velocity on the boundaries of the trajectory can be fixed at will through the boundary conditions (14), but the acceleration on these boundaries is unfortunately imposed by the shape of the solution.

4.2 Numerical approximation of the solution in the general case

In the general case the simple dynamics (8) of the previous section turns into [KD99]:

$$\Gamma = M(q)\ddot{q} + N(q,\dot{q})\dot{q} + G(q) + H(\dot{q}) = f(q,\dot{q},\ddot{q}), \tag{23}$$

with $M(q)$ the inertia matrix, $N(q,\dot{q})$ the matrix of centrifugal and coriolis effects, $G(q)$ the gravity effects and $H(\dot{q})$ the friction. This dynamics is much more complex than the previous one and an analytical solution to this general trajectory optimization problem might be out of reach. We are going therefore to look for a numerical approximation of the solution.

The problem of trajectory generation is generally considered in robotics as a problem of optimal control where the state equation is in the form:

$$\ddot{q} \quad = \quad F(q,\dot{q},\Gamma). \tag{24}$$

Expressing the dynamics of the system in this form implicitely introduces then the idea to solve this dynamics in order to obtain the trajectory $(q(.),\dot{q}(.))$ for a given command $\Gamma(.)$. The gradient of the cost function needs therefore to be calculated with adjoint variables methods [Lem95]. There exist then two general classes of methods to solve this kind of problem [vS98] [Bet97]: indirect methods based on the application of the Maximum Principle of Pontryagin (using adjoint equations), generally considered to be very precise but very sensitive to initial conditions, and direct methods based on the discretization of both the trajectory $(q(.),\dot{q}(.))$ and the command $\Gamma(.)$ (without using adjoint equations), leading to a classical non linear optimization problem and considered to be less precise but less sensitive to initial conditions.

However, it may be an error to express the dynamics (23) as in equation (24): with dynamics expressed as in (23) the problem of trajectory generation appears directly as a problem of calculus of variations. The dynamics (23) can be directly integrated in the different constraints of our problem which becomes then:

$$\min_{(q,\dot{q})} t_f$$

$$\int_0^{t_f} A\, f(q,\dot{q},\ddot{q})^2 + B\,\dot{q}^2\, dt + t_f\,(\gamma + T_{amb} - T_{max}) \quad \leq \quad 0. \tag{25}$$

There are two ways of solving this problem: it is possible to calculate the optimality conditions similarly to what we've done in the previous section, but that may lead to set of complex non-linear ordinary differential equations with boundary conditions that may not be easy to solve. Another way is to consider a discrete approximation of the trajectory $q(t)$:

$$\tilde{q}(t) \quad = \quad S(p,t) \tag{26}$$
$$\dot{\tilde{q}}(t) \quad = \quad \dot{S}(p,t) \tag{27}$$
$$\ddot{\tilde{q}}(t) \quad = \quad \ddot{S}(p,t) \tag{28}$$

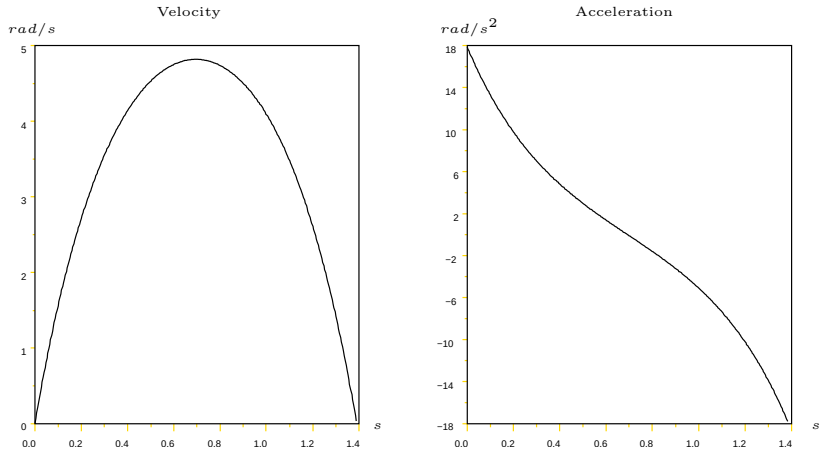

Figure 5: Optimal velocity and acceleration profiles

with S an interpolation function at least of class C^2, and p the parameters of the interpolation. The dynamics (23) takes then the following form:

$$\tilde{\Gamma}(t) = f(S(p,t), \dot{S}(p,t), \ddot{S}(p,t), t) = \tilde{f}(p,t) \tag{29}$$

Discretizing the trajectory $q(t)$ has several advantages: a small number of discretization parameters (the same ones for q, $\dot{q}$, $\ddot{q}$ and Γ), a finite dimensional problem implying a simpler calculation of the gradients, and the use of improved non-linear optimization algorithms such as Sequential Quadratic Programs (SQP) which are presently the most efficient algorithms for such problems. Note that similar choices have been tested in [vS98] with good results.

4.2.1 Definition of the optimization problem

We aren't interested here in optimizing the geometric path of the trajectory and we focus rather on the optimization of the velocity profile of the trajectory along a specified geometric path. The first step consists therefore in defining the curvilinear abscissa $\lambda : [0, t_f] \rightarrow [0, 1]$ (also called the time law) and the geometry function $Q : [0, 1] \rightarrow \mathbb{R}^6$ which transforms this curvilinear abscissa into an articular position, both functions being at least of class C^2 such that:

$$q(.) = Q(\lambda(.)) \tag{30}$$

$$\dot{q}(.) = \frac{dQ}{d\lambda}(\lambda(.))\dot{\lambda}(.) \tag{31}$$

$$\ddot{q}(.) = \frac{d^2Q}{d\lambda^2}(\lambda(.))\dot{\lambda}^2(.) + \frac{dQ}{d\lambda}(\lambda(.))\ddot{\lambda}(.) \tag{32}$$

The second step consists in discretizing this curvilinear abscissa: there exist various techniques of discretization but the most usual are polynomial splines [vS98] [LCL83] [LLO91] [TP87]. Since the time law must be at least of class C^2, we will use cubic splines and more precisely we will compute them here as in [LLO91]. The spline is defined as shown in figure 6, with $\lambda(t_i) = \Lambda_i$ for $1 \leq i \leq n$, $t_1 = 0$ and $\Lambda_1 = 0$, $t_n = t_f$ and $\Lambda_n = 1$. Since $t_n = t_f$ is variable whereas $\Lambda_n = 1$ is fixed, we will consider that all the $\{t_i\}_{(1 \leq i \leq n)}$ are variable whereas all the $\{\Lambda_i\}_{(1 \leq i \leq n)}$ are fixed, leading to a non-uniform spline.

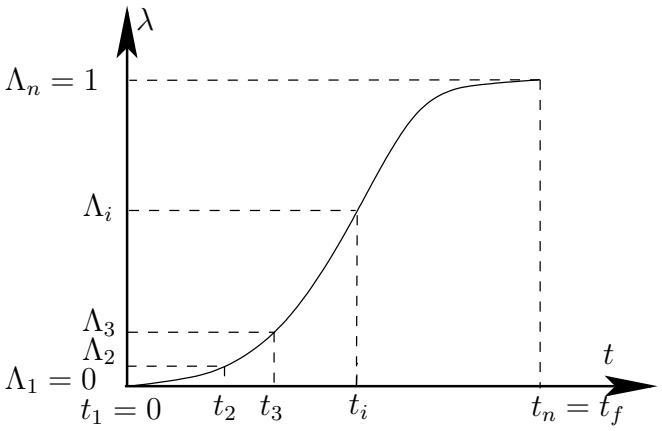

Figure 6: Discrete time law

Following [LLO91], our strategy to define this cubic spline is to impose the continuity of the velocity and the acceleration at the nodes t_i, and to fix the velocity on the boundaries. We use the intermediate variables $\{\ddot{\Lambda}_i\}_{(1 \leq i \leq n)}$ to fix the acceleration at each knot. Each of the cubic polynomials $\lambda_j(t) = \lambda(t)$ for $t \in [t_j, t_{j+1}]$ constituting the spline can be written in terms of the $\ddot{\Lambda}_j$ and the h_j:

$$\lambda_j(t) = \frac{(t_{j+1} - t)^3}{6h_j}\ddot{\Lambda}_j + \frac{(t - t_j)^3}{6h_j}\ddot{\Lambda}_{j+1} + \left(\frac{\Lambda_{j+1}}{h_j} - \frac{h_j\ddot{\Lambda}_{j+1}}{6}\right)(t - t_i) + \left(\frac{\Lambda_j}{h_j} - \frac{h_j\ddot{\Lambda}_j}{6}\right)(t_{i+1} - t). \tag{33}$$

The continuity of the velocity is satisfied then by solving a linear system that leads to the computation of the $\{\ddot{\Lambda}_i\}_{(1\leq i\leq n)}$:

$$C(h)\ddot{\Lambda} = d(h) \tag{34}$$

with:

$$C(h) = \begin{pmatrix} 2h_1 & h_1 & & & & 0 \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & & & & h_{n-1} & 2h_{n-1} \end{pmatrix}$$

$$d(h) = \begin{pmatrix} 6\left(\frac{\Lambda_2-\Lambda_1}{h_1} - v_1\right) \\ 6\left(\frac{\Lambda_3-\Lambda_2}{h_2} - \frac{\Lambda_2-\Lambda_1}{h_1}\right) \\ \vdots \\ 6\left(\frac{\Lambda_n-\Lambda_{n-1}}{h_{n-1}} - \frac{\Lambda_{n-1}-\Lambda_{n-2}}{h_{n-2}}\right) \\ 6\left(v_n - \frac{\Lambda_n-\Lambda_{n-1}}{h_n}\right) \end{pmatrix}$$

with $h_i = t_{i+1} - t_i$ the time intervals between knots, and $h = (h_1, h_2, ..., h_{n-1})^T$ the new set of parameters for the optimization procedure. After solving the linear system (34), the spline is totally determined by the $\{\Lambda_j\}_{1\leq j\leq n}$, the velocity on the boundaries v_1 and v_n and the time intervals $\{h_i\}_{1\leq i\leq n-1}$. Note that the matrix C is non singular here since it is diagonally dominant, and there are efficient numerical methods to invert such tridiagonal matrices.

Now, the cost function can be simply expressed by a linear function of the parameters, $t_f = \sum_{i=1}^{n-1} h_i$. The thermal model (1) leads to the following constraint:

$$A\int_0^{t_f} \Gamma^2(t)\,dt + B\int_0^{t_f} \dot{q}^2(t)\,dt + t_f\,(\gamma + T_{amb} - T_{max}) \leq 0, \tag{35}$$

with T_{amb} the ambiant temperature and T_{max} the maximal authorized temperature. The integral temperature constraint (35) can be estimated then with a trapezoidal approximation, precise enough in our case as we will see in the section 6:

$$\frac{1}{2}\sum_{j=0}^{n-1}\left(A(\Gamma(t_j)^2 + \Gamma(t_{j+1})^2)h_j + B\left(\dot{q}(t_j)^2 + \dot{q}(t_{j+1})^2\right)h_j\right) + t_f(\gamma + T_{amb} - T_{max}) \leq 0. \tag{36}$$

4.2.2 Optimization algorithm

The original trajectory optimization problem has been transformed into a minimization of a linear function subject to non linear constraints. Newton methods such as Sequential Quadratic Programming (SQP) are presently the most efficient ones to solve this kind of problem [BGLS03]. These methods need the gradients of both the cost function and the constraints. The gradients can be calculated numerically with finite differences methods, but that severely impedes the convergence of the SQP. Symbolic or automatic differentiation methods [DKH02] can be used also, but in our problem the geometric path is fixed and the dynamics (23) can be formulated very simply as [BDG85] [Hol84] [Žla96]:

$$\Gamma = m(\lambda)\ddot{\lambda} + c(\lambda)\dot{\lambda}^2 + f(\dot{\lambda}) + g(\lambda) \tag{37}$$

where m, c, g and f are vectors which respectively represents inertias, centrifugal and coriolis effects, gravity and friction. Since the constraints are evaluated at points $\{\Lambda_j\}_{1\leq j\leq n}$ which don't depend on the variables $\{h_i\}_{1\leq i\leq n-1}$, the gradient of the dynamic model is:

$$\frac{\partial\Gamma_j}{\partial h_k} = m(\Lambda_j)\frac{\partial\ddot{\Lambda}_j}{\partial h_k} + 2\,c(\Lambda_j)\frac{\partial\dot{\Lambda}_j}{\partial h_k}\dot{\Lambda}_j + \frac{\partial f}{\partial h_k}(\dot{\Lambda}_j) \tag{38}$$

Since we suppose that we optimize trajectories where the velocity has always the same sign, we can rely on a simple expression of the gradient of the Coulomb and viscous friction:

$$\frac{\partial f}{\partial h_k}(\dot{\Lambda}_j) = F_v \frac{dQ}{d\lambda} \frac{\partial \dot{\Lambda}_j}{\partial h_k} \tag{39}$$

The calculation of the gradient of the dynamic model amounts then to calculating $\frac{\partial \dot{\Lambda}_j}{\partial h_k}$ and $\frac{\partial \ddot{\Lambda}_j}{\partial h_k}$, what is trivial here through equation (34).

The last important point is the initialization of the optimization process: to help its convergence, we must choose a first iterate as close as possible to the optimal solution and satisfying all the constraints. From a practical point of view, we generate a BANG-zero-BANG profile, and we use a dichotomy technique to improve the first iterate by testing the constraints: the duration of the movement is stretched if any constraint is violated, compressed otherwise.

Note that throughout this section, we have always considered single point to point motions with a constant sign of the velocity. More complex motions can be built by gluing together such motions, the global constraint on the temperature being taken care of then at the level of the global optimizer of figure 3. This whole procedure will be tested and validated in section 6.

5 Global optimization of robot applications with hardware in the loop

The description of the general optimization algorithm in section 3 has shown the need to solve the global problem (7) with a cost function and inequality constraints which need to be evaluated from data directly measured on the robot, what appeared to be the only way to take into account the unmodelized part of the whole robotic cell.

A similar scheme can be found in Iterative Learning Control methods, when a robot repeatedly attempts to execute a prescribed task while an adaptation algorithm successively improves the control system's performance from one trial to the next by updating the control input based on the error signals from previous trials [Lon00] [Hor93]. But such methods can't be easily applied to problems with global criteria and constraints, such as the cycle time and the temperature constraints that we need to deal with here. More than that, the tasks that we consider here aren't exactly cyclic, what is generally a strict requirement for applying such methods successfully.

The difficulty in such algorithms is to deal with noisy data, with gradients of the criterion and constraints that don't exist or can't be obtained easily and efficiently: we must use therefore optimization methods without derivatives.

5.1 Unconstrained optimization without derivatives

Since we focus on optimization methods without derivatives, direct search methods as discussed in [Pow98] and [CST97] are to be looked for. The Nelder-Mead simplex method is one of the most frequently used algorithm in optimization without derivatives, but it doesn't converge in some cases and suffer from inefficiency when the dimension of the problem is too large. Other methods such as simulated annealing or genetic algorithms suffer from similar limitations [Pow98].

A real improvement in direct search methods has been obtained when Powell described a method for solving non-linear unconstrained minimization problems based on the use of conjugate directions [Pow64]. The main idea of this proposal is that the minimum of a positive definite quadratic form can be found by performing at most n successive linear searches along mutually conjugate directions. Then he proposed (independently of [Win73]) to use the available objective values for building a quadratic model. This model is assumed to be valid in a neighborhood of the current iterate, which is described as a trust region, whose radius is iteratively adjusted. The model is then minimized within this trust region, hopefully yielding a point with a low function value. The difficulty in this kind of algorithms is that the set of interpolation points must have certain geometric properties. Powell proposed therefore an algorithm where the set of interpolation points is updated in a way that preserves its geometric properties in the sense that the differences between points of this set are guaranted to remain sufficiently linearly independent.

We can find now two efficient algorithms for optimization without derivatives: Derivative Free Optimization (DFO) [CST97] from Conn, Scheinberg and Toint and NEWUOA from Powell [Pow04b]. They are both based on the identification of a quadratic model of a function $\Phi : \mathbb{R}^n \to \mathbb{R}$ at each iteration of a numerical algorithm. For the k^{th} iteration the model is:

$$m_k(p_k + s) = \Phi(p_k) + < g_k, s > + \frac{1}{2} < s, H_k s > \tag{40}$$

with $< ., . >$ the standard euclidian scalar product, g_k a real vector and H_k a symmetric matrix. g_k and H_k are estimated by interpolating a set Y of points:

$$\forall\, y \in Y,\ m_k(y) = \Phi(y). \tag{41}$$

Note that the number of elements of Y must be:

$$p = \frac{1}{2}(n+2)(n+1)$$

in order to estimate all the parameters of the model. The interpolation set Y is updated at each iteration in order to preserve its geometric properties. The number of elements of Y is an important point of difference between DFO and NEWUOA, since NEWUOA actually needs a limited number p of elements in Y:

$$(n+2) \leq p \leq \frac{1}{2}(n+1)(n+2).$$

The important point here is that the evaluation of the original function Φ can be a costly process, especially in our case where a whole application cycle is required each time: the NEWUOA algorithm allows then to reduce strongly the total number of such evaluations with respect to the DFO algorithm, reducing therefore strongly the total cost of the optimization. The number of elements of Y is decreased since NEWUOA identifies the quadratic model by using the equations of interpolation (41) and by minimizing the Frobenius norm of the difference between H_k and H_{k-1} [Pow04a]. When the quadratic model is identified, NEWUOA finds its minimum within the trust region, the cost function is then evaluated at this point, if the decrease of the cost function is worse than the decrease predicted by the model, the radius of the trust region is scaled down, the radius is not changed otherwise. The minimization algorithm by successive quadratic approximations is summarized in Table 2:

(i)	Initialize the set Y, the radius of the trust region and the first iterate
(ii)	build the quadratic model
(iii)	Minimize this model within the trust region
(iv)	Update the set of interpolation Y
(v)	Update the radius of the trust region
(vi)	Update the current iterate and go to step (ii)

Table 2: Minimization algorithm by successice quadratic approximations

The algorithm finishes when the radius of the trust region attains a lower bound fixed by the user. NEWUOA appears to be one of the best algorithms available today for minimizing a cost function without derivatives. But NEWUOA doesn't deal with constraints, and our problem is subject to constraints. We need therefore to focus now on how to take into account inequality constraints when the derivatives are not available.

5.2 Penalty methods in non-linear programming

Historically, the earliest developments in non-linear programming were sequential minimization methods with the use of penalty and barrier functions, what represents a global approach to non-linear programming as opposed to local methods based on the linearization of the constraints. Since we don't have access to the gradients of our criterion and constraints, we must use such sequential methods which can be separated in two classes: penalty methods and exact penalty methods.

5.2.1 Inexact penalty functions

Historically, Courant developped a method in order to take into account equality constraints which can be adapted easily to inequality constraints by the use of the max function [Fle87]. The point is to consider the penalized function:

$$\Phi(p,\sigma) = t_c(p) + \sigma \sum_{i=1}^{m} [\max(c_i(p),0)]^2. \tag{42}$$

This function can be used in an iterative scheme such as in Table 3.

(i)	Choose a fixed sequence $\{\sigma^{(k)}\}$, for example $\{1, 10, 10^2, 10^3, ...\}$
(ii)	For each σ^k, find a local minimizer $p(\sigma^k)$ to $\min_p \Phi(p, \sigma^k)$
(iii)	Terminate when $\max(c_i(p), 0)$ is sufficiently small

Table 3: Iterative scheme for inexact penalty functions

Proofs of convergence of such a method are available in [Fle87]. There exist other penalty functions such as barrier functions which are usually used in interior point methods. Since these barrier functions, usually inverse or logarithm functions, are not defined when the constraints are active, we won't use this kind of penalty functions.

One experience in the next section will be devoted to compare an exponential penalty function:

$$\Phi(p, \sigma) = t_c(p) + \sum_i \sigma_i e^{c_i(p)} \tag{43}$$

to a second class of penalty methods: the exact penalty functions.

5.2.2 Exact penalty functions and the augmented Lagrangian method

For a large enough σ, the penalization:

$$\Phi(p, \sigma) = t_c(p) + \sigma \sum_i \| \max(c_i(p), 0)\|_1 \tag{44}$$

is exact, that is to say the minimum of this function is the same as the optimum of problem (7), as proved in [BGLS03] or [Fle87].

An advantage is the exactness of the solution to the original problem for a finite σ, with no need to iterate on the value of σ. But this penalty function (44) is not differentiable at the optimum, and since minimization algorithms without derivatives are not designed for such non differentiabilities, that can lead to severe troubles: we should look for another solution.

Lagrangian methods [HU01] can also be seen as exact penalty methods when the function to minimize and the constraints are convex. In our case, we don't know whether the functions $t_f(p)$ and $c_i(p)$ are convex or not, but they are most probably not. The augmented Lagrangian method deals with this kind of problem, with a Lagrangian actually augmented with a quadratic form of the constraints, tending to create a convex basin around the optimum of problem (7) [BGLS03]. When Powell discovered this method [Pow69], he exposed this technique in a more intuitive way. Actually, to get the minimum of problem (7) with the Courant penalty method, σ must tend to infinity. To avoid this problem, we could add a shift parameter θ_i for each constraint :

$$\Phi(p, \theta, \sigma) = t_c(p) + \frac{1}{2} \sum_i \sigma_i (\max(c_i(p) - \theta_i, 0))^2. \tag{45}$$

More details and illustrative examples are available in [Fle87].

By using the augmented Lagrangian method, a constrained minimization problem is translated into a sequence of unconstrained problems (with $\lambda_i = \theta_i \sigma_i$) as shown in Table 4: the step (ii) of Table 4 is the non trivial

(i)	$\lambda \leftarrow \lambda^{(1)}$, $\sigma \leftarrow \sigma^{(1)}$, $k \leftarrow 0$, $\|\nabla\Psi^{(0)}\|_\infty \leftarrow \infty$
(ii)	Find the minimizer $\mathbf{p}(\lambda, \sigma)$ of $\Phi(\mathbf{p}, \lambda, \sigma)$ and denote $\mathbf{c} = \mathbf{c}(\mathbf{p}(\lambda, \sigma))$
(iii)	If $\|[-\max(c_i, \frac{\lambda_i}{\sigma_i})]_{i=1..m}\|_\infty > \frac{1}{4}\|\nabla\Psi^{(k)}\|_\infty$ then : $\forall i$, if $\|c_i\| > \frac{1}{4}\|c^{(k)}\|_\infty$ then $\sigma_i \leftarrow 10\sigma_i$ go to step (ii)
(iv)	$k \leftarrow k+1$, $\lambda^{(\mathbf{k})} \leftarrow \lambda$, $\sigma^{(\mathbf{k})} \leftarrow \sigma$, $\mathbf{c}^{(\mathbf{k})} \leftarrow \mathbf{c}$
(v)	$\forall i = 1..m, \lambda_i \leftarrow \lambda_i^{(k)} - \max(\sigma_i c_i^{(k)}, \lambda_i^k)$ and $\|\nabla\Psi^{(k)}\|_\infty \leftarrow \|[-\max(c_i, \frac{\lambda_i}{\sigma_i})]_{i=1..m}\|_\infty$

Table 4: Numerical scheme using the augmented Lagragian method

step of the algorithm and can be solved by using a derivative free algorithm like NEWUOA. This algorithm has several advantages :

- derivatives of the cost function and the constraints never appear in this scheme,
- this is an exact penalty method

- the usual ill conditionning of penalty methods when $\sigma \to \infty$ doesn't appear here since the σ_i and θ_i are finite numbers.

Since we don't use derivatives, the convergence of step (ii) of the algorithm in Table 4 can take a long time, and the convergence of the whole algorithm can really be much longer than inexact penalty methods. Moreover, this algorithm doesn't guarantee that the constraints won't be violated during the iterations (this is not an interior point algorihtm). These are the two major disavantages of our approach. We will be able to quantify the impact of these disavantages on our on-line trajectory optimizer during the experiments with industrial applications in the next section.

Remark 5.1. *This section describes in fact a general methodology for optimizing robotic applications with hardware in the loop which is not particularly bound to solving our original problem (5). We will already see in the next section a small variation on this original problem, where the sub-problems (6)-(5) will be replaced by a classical BANG-zero-BANG trajectory generator, with the parameters p being therfore the maximum acceleration, deceleration and velocity. Even more general optimization problems, with different costs to optimize and different constraints to comply with, could be tackled efficiently by applying the same identical methods*

6 Numerical and experimental validation

It is necessary now to validate the three algorithms developped in the previous sections: the optimal profile generator of section 4, the global optimizer of section 5 and the complete algorithm described in Figure 3 of section 3. We first validate the optimal profile generator and the global optimizer separately in sections 6.2 and 6.3, all these experiences lead to a comparison between all the optimized profiles in section 6.4. All the tests will be based on three robot applications which will be described first of all in section 6.1.

6.1 Description of the robot tasks to optimize

In order to verify the convergence of the optimal profile generator of section 4, we will apply it to a simple case, a movement of the first axis of a Stäubli Rx90 in flat configuration (Figure 7) from -2.26 rad to $+2.26$ rad with a pause of $0.5\,s$ at the end position. The analytical solution to this simple robot task has already been described in Figure 5: we will be able therefore to verify the precision of the numerical scheme by direct comparison with this analytical solution. The robustness of the convergence of this optimal profile generator will be tested then by applying it to a real industrial application, the pick and place application shown in Figure 8.

Figure 7: Flat configuration of a Stäubli Rx90

In order to test the global optimizer of section 5 independently from the optimal profile generator of section 4, we will apply it first with a BANG-zero-BANG profile generator: the parameters p that need to be optimized are therefore the maximum acceleration, velocity and deceleration between each pair of points. This will be applied to the pick and place application of Figure 8 and a load/unload application described in Figure 9. This load/unload application implies 4.5 times more parameters to optimize than the pick and place application, what will allow testing the applicability of the global optimizer on large real-life problems.

Applying independently the optimizer of section 5 and the optimal profile generator of section 4 on the same pick and place application of Figure 8 will allow us to compare optimized BANG-zero-BANG profiles to truly optimal profiles, allowing to quantify the contribution of the latter with respect to the former.

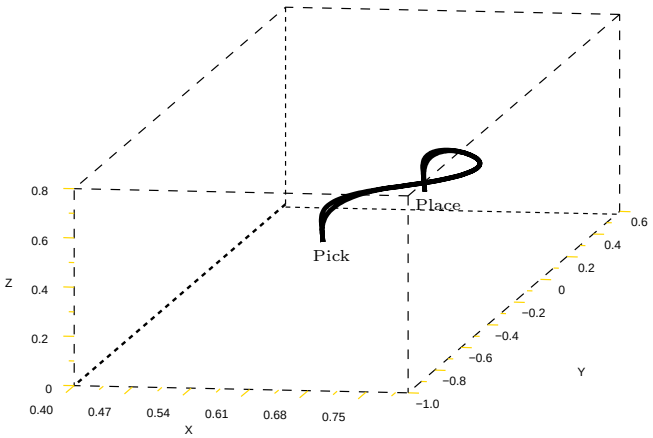

Figure 8: Geometric trajectory of a manipulator robot during a pick and place application

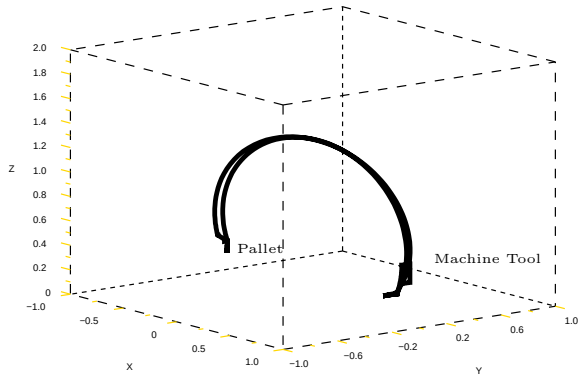

Figure 9: Geometric trajectory of a manipulator robot during a load/unload application

For testing the complete algorithm of Figure 3 with two levels of optimization, only simulations have been realized since we didn't have access to a real robot at that moment for experiments. It is tested first of all on the same simple application as before, the movement of the first axis from $-2.26\,rad$ to $2.26\,rad$ with a pause of $0.5\,s$, but under real working conditions, that is with a total cycle time and the corresponding motor torques and speeds not known in advance. The pause of $0.5\,s$ and its impact on the thermal constraint isn't modelized here but discovered through measures realized on a simulated robotic cell.

The pick and place application of Figure 8 is considered then, split into 9 point to point motions in order to validate the ressource decomposition method introduced in section 3.

6.2 Optimal profile generator

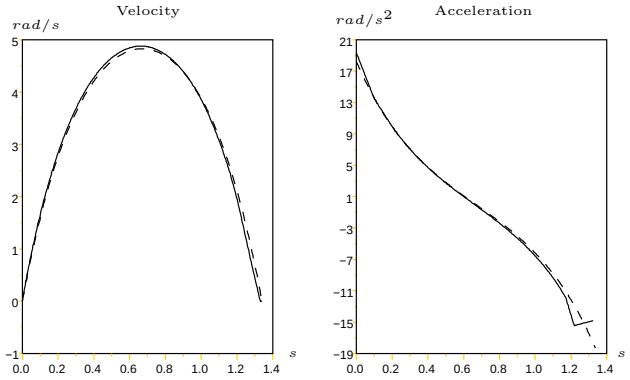

Figure 10: Analytical (dashed curve) and numerical (plain curve) optimal profiles

When applied to the simple task of Figure 5, the numerical algorithm described in section 4.2 converges to the solution showed in Figure 10, with an optimal time $t_f = 1.35\,s$ (using the Feasible Sequential Quadratic Programming (FSQP) algorithm [LZT97] to solve the underlying non-linear optimization problem). We can observe that it is very close to the analytical solution found in section 4.1. The very small difference between these two solutions can be identified to be solely due to the discretization process (26)-(28). For the same reason, the approximate computation in (36) of the constraint (35) appears to slightly underestimate the limiting temperature constraint in this specific case, allowing a faster solution here than the truly optimal solution described in section 4.1, with only $1.35\,s$ of cycle time instead of $1.38\,s$ in section 4.1.

More generally, this algorithm has been observed to converge properly as soon as the constraints are satisfied from the beginning of the optimization process, as soon as the first iterate is a feasible point. This condition appeared to be of great importance to obtain this convergence: the initialization described at the end of section 4.2.2 appears therefore to be a key point for the robustness of the whole numerical algorithm.

This trajectory has been executed then on a real Stäubli Rx90 robot without filtering, in spite of the discontinuities of the acceleration at the boundaries. The stabilized temperature measured after 6 hours reaches

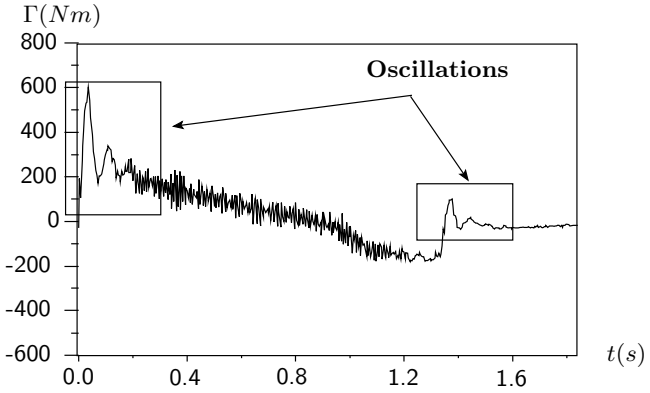

Figure 11: Measured torque

the prescribed limit with only 3% of error. Both the dynamic and the temperature models appear therefore to allow very precise predictions. Note however that oscillations appear in Figure 11 which have not been predicted. These oscillations are due to the discontinuities of the acceleration on the boundaries that excite the vibration modes of the robot (fixing the acceleration at the boundaries should solve very simply this problem).

6.3 Global optimization with BANG-zero-BANG profiles

Then we test the global optimizer of Figure 3 with a BANG-zero-BANG profile generator. Three different experiments are realized in order to:

- compare the exponential penalty function (43) with the augmented Lagrangian (45),
- test the robustness to task changes,
- test the influence of a large number of trajectory parameters,

The comparison of the exponential penalty function (43) with the augmented Lagrangian (45) is realized on the optimization of the pick and place application of Figure 8, with weighting coefficients initialized as in Table 5. Figure 12 shows an identical evolution in both cases in the beginning (in the boxed area): this is due to the

Exponential penalty function	Augmented Lagrangian penalty function
$\sigma^1 = [50, ..., 50]^T$	$\sigma^{(1)} = [100, ..., 100]$ $\lambda^{(1)} = [0, ..., 0]$

Table 5: Initial values of the penalty coefficients.

initialization of the quadratic approximation of the cost function which is always realized in the same systematic way by the NEWUOA algorithm. We can observe then in both cases a decrease of the cycle time while the constraints rise up to their limit, with convergence in less than half an hour. We can observe that the use of an exponential penalty function implies a milder management of the temperature constraints, but both methods lead to an equivalent cycle time: it seems therefore difficult to make a clear choice between them. Note also that the constraints can be violated temporarily during the convergence process of both methods, as can be seen in Figures 12 and 13: this shouldn't be problematic as long as the only constraint being considered is a stabilized temperature, but this can be an important detail in other cases.

The experiment for testing the robustness of this optimization method to task changes consists in executing the same pick and place application as before but carrying a load of 6 Kg. The torques and the velocities of the actuators and therefore the stabilized temperature and the cycle time are altered: without giving any specific information to the algorithm, a convergence to a different slower solution can be observed in Figure 13. The algorithm automatically takes into account the changes in the robot task thanks to the use of data directly recorded by sensors on the robot in order to find an appropriate solution.

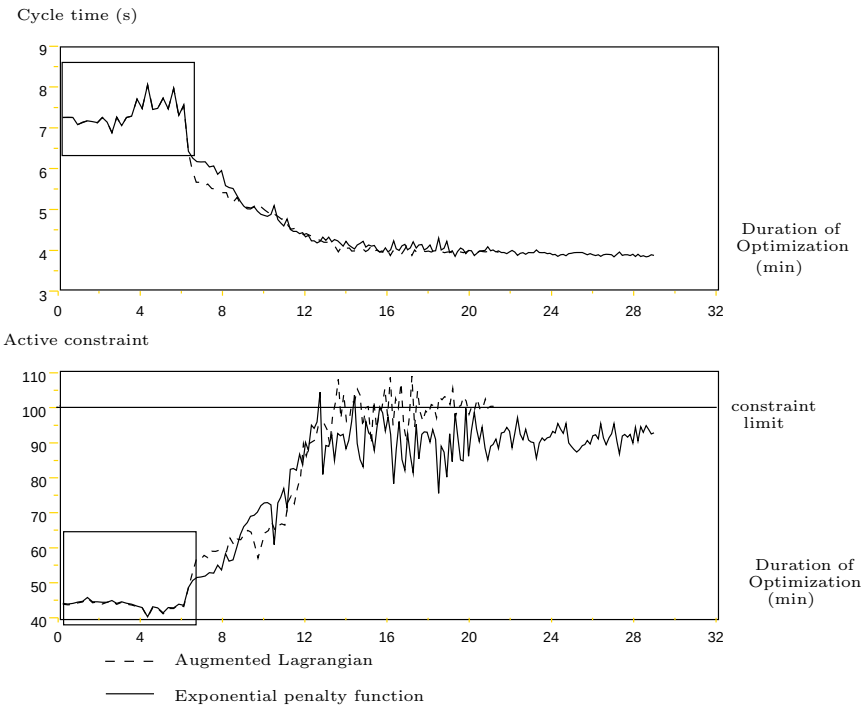

Figure 12: Convergence of the algorithm for the pick and place application without load.

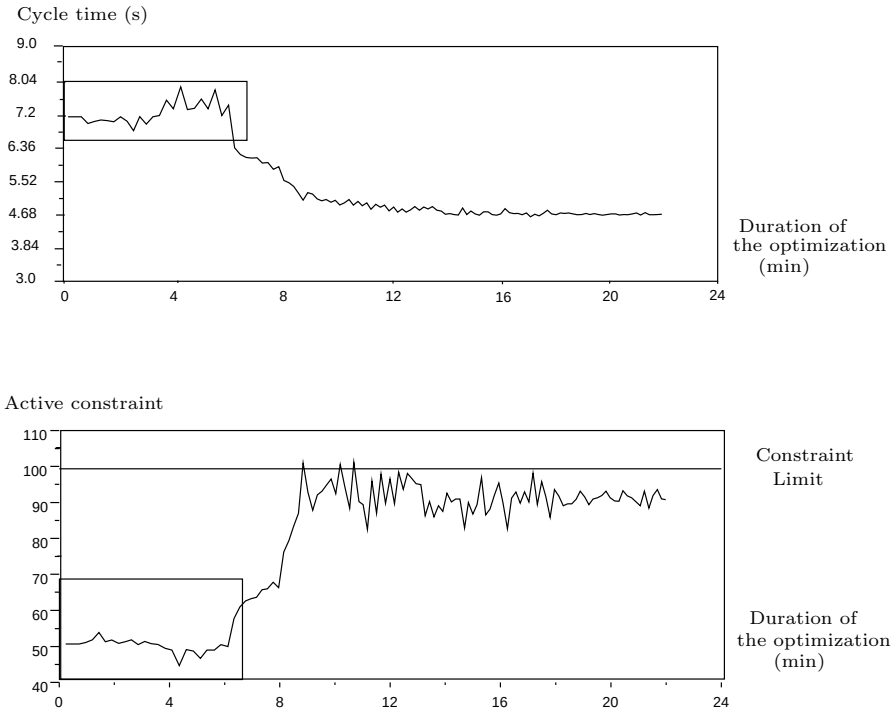

Figure 13: Convergence of the algorithm for the pick and place application with a load of 6Kg

The experiment for testing the influence of a large number of parameters consists in the load/unload application of Figure 9 with 54 parameters instead of only 12 for the pick and place application of Figure 8. A minimum is reached after 5 hours of optimization instead of 30 minutes earlier, as shown in Table 6. Table 7 shows that the improvement of the performance of the robot with respect to the nominal constructor settings is less than for the previous application, but still of 8%. Note that 5 hours for optimizing a robotic application isn't long when this application is going to be executed repeatedly 8% faster for months or years. The proposed algorithm appears therefore to be well adapted to the optimization of a complex industrial applications.

Application	Number of trajectory parameters	Number of cycles	Duration of the optimization
Pick & Place	12	600	30 min
Pick & Place with load	12	400	30 min
Load/unload	54	1800	5 h

Table 6: Optimization time before reaching convergence.

Application	Nominal cycle time	After Optimization	Gain
Pick & Place	6.52s	3.89s	**40%**
Pick & Place with load	6.52s	4.66s	**28%**
Load/unload	8.12s	7.44s	**8.4%**

Table 7: Gain of cycle time using the global optimizer.

6.4 Comparison between the different optimized profiles

We can quantify then the increase of performance when using the truly optimal velocity profiles with respect to optimized BANG-zero-BANG profiles on the simple task of Figure 5 and the pick and place application of Figure 8. Table 8 shows that the optimized BANG-zero-BANG profiles are already 40 to 50% faster than the nominal profiles suggested by the constructor, but the truly optimal velocity profiles are still 3 to 6% faster, what appears still as a significant increase in productivity on an industrial set-up.

Table 8 also shows that the profiles optimized by the complete algorithm in real industrial conditions are only 0.5% slower than the truly optimal velocity profiles and 1.5 to 4% faster than the optimized BANG-zero-BANG profiles: it allows therefore a significant increase of productivity with respect to classical trajectory generators.

Application	Nominal	optimized BANG-zero-BANG profile	optimal velocity profile	Complete algorithm
Simple task	7.91s	3.76s	3.68s	3.70
Pick & Place	6.52s	3.89s	3.72s	3.74

Table 8: Comparison of the cycle time resulting from different methods of optimization and the nominal profile.

7 Conclusion

Numerous works on trajectory optimization in robotics have explored different techniques to find optimal trajectories [Bes92] [LLO91] [Hol84] [BDG85] [LCL83]. They usually only focus on algorithmic and numerical aspects, but they don't use precise models of the real physical limitations such as the thermic one considered here, and they don't take into account the integration of the robots in an industrial robotic cell which is usually very imperfectly modelized. These two aspects can't be neglected if we want to reach the maximum performances of a robot in real industrial working conditions, i.e. when the robot is integrated in a complex robotic cell.

We have pointed out here in the first section that the structure of a robotic application isn't directly compatible with this optimization problem we have to deal with, but it has a great property: it is decomposable. This property allows to split the optimization in two levels. We have derived then two algorithms for these two levels using different optimization techniques:

- an optimal profile generator which uses an optimization based on precise models of the robot and only needs a limited view of the robot task, solving a problem of calculus of variations using a direct method,
- a global optimization algorithm with hardware in the loop which allocates energetic ressources to each point to point motion taking into account all the imperfectly modelized interactions between the robot and the cell, based on optimization techniques without derivatives and on penalty methods to take into account the constraints.

The experimental results obtained on a real Stäubli Rx90B manipulator robot with these algorithms are excellent! Most importantly, they are able to adapt the behavior of the robot to changes in the task without any intervention from the operator. On top of that, the resulting optimal velocity profiles appear to be 5 to 10% faster than classical BANG-zero-BANG profiles, inducing a dramatic increase of productivity of the whole robotic cell. This work is protected by patent applications.

References

[BDG85] James Bobrow, S. Dubowsky, and James Gibson. Time optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 1985.

[Bes92] Yasmina Bestaoui. On line reference trajectory definition with joint torque and velocity constraints. *The International Journal of Robotics Research*, 1992.

[Bet97] John Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 1997.

[BGLS03] Joseph-Frédéric Bonnans, Jean-Charles Gilbert, Claude Lemaréchal, and Claudia Sagastizabal. *Numerical Optimization : Theoritical and Practical Aspects*. Springer, Collection, 2003.

[BH75] Arthur Bryson and Yu-Chi Ho. *Applied Optimal Control, Optimization, Estimation and Control*. Taylor and Francis, 1975.

[BM03] Geoffrey Biggs and Bruce MacDonald. A survey of robot programming systems. *Proceedings of the Australasian Conference on Robotics and Automation in Brisbane, Australia*, 2003.

[cL89] Matsuhita Electric Ind. co Ltd. Controller for industrial robot. *Japan Patent JP1020990*, 1989.

[Cor01] Denso Corp. Motor controller. *Japan Patent JP2001292586*, 2001.

[CST97] A.R. Conn, K. Scheinberg, and Ph.L. Toint. Recent progress in unconstrained nonlinear optimization without derivatives. *ISMP97, Lausanne*, 1997.

[DKH02] Axel Durrbaum, Willy Klier, and Hubert Hahn. Comparison of automatic and symbolic differentiation in mathematical modeling and computer simulation of rigid-body systems. *Multibody Systems Dynamics*, 2002.

[Fle87] Roger Fletcher. *Practical Methods of Optimization, Second Edition*. John Wiley and Sons, 1987.

[Hol84] John Hollerbach. Dynamic scaling of manipulator trajectories. *Journal of Dynamic Systems, Measurement, and Control*, 1984.

[Hor93] Roberto Horowitz. Learning control of robot manipulators. *ASME Journal of Dynamic Systems Measurement and Control 50th Anniversary Issue*, 1993.

[HU01] Jean-Baptiste Hiriart-Urruty. *Optimisation*. Que sais-je ?, 2001.

[KD99] Wisama Khalil and Etienne Dombre. *Modélisation, identification et commande des robots*. Hermes Science Publications, 1999.

[LCL83] Chun-Sin Lin, Po-Rong Chang, and J.Y.S. Luh. Formulation and optimization of cubic polynomial joint trajectories for industrial robots. *IEEE Transactions on Automatic Control*, 1983.

[Lem95] Claude Lemaréchal. Optimisation. *Les techniques de l'ingénieur : Automatique*, 1995.

[LLO91] Alessandro De Luca, Luca Lanari, and Giuseppe Oriolo. A sensitivity approach to optimal spline robot trajectories. *Automatica*, 1991.

[Lon00] Richard W. Longman. Iterative learning control and repetitive control for engineering practice. *International Journal of Control, Vol. 73, No 10, p.930-954*, 2000.

[Ltd88] Kobe Steel Ltd. Method for foreseeing working limit of teaching playback type robot. *Japan Patent JP63126009*, 1988.

[Ltd95] Fanuc Ltd. Displaying method for duty of industrial robot. *Japan Patent JP7087787*, 1995.

[LZT97] Craig Lawrence, Jian Zhou, and André Tits. User's guide for cfsqp version 2.5 : A c code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Technical report, Electrical Engineering Department and Institute for Systems Research, University of Maryland, 1997.

[Pin93] Enid Pinch. *Optimal Control and the calculus of Variations*. Oxford Science Publications, 1993.

[Pow64] Mickael Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal*, 1964.

[Pow69] Mickael Powell. A method for nonlinear constraints in minimization problems. 1969.

[Pow98] Mickael Powell. Direct search algorithms for optimization calculations. 1998.

[Pow04a] Mickael Powell. Least frobenius norm updating of quadratic models that satisfy interpolation conditions. *Mathematical Programming, vol. 100*, 2004.

[Pow04b] Mickael Powell. The NEWUOA software for unconstrained optimization without derivatives. *40th Workshop on Large Scale Nonlinear Optimization (Erice, Italy)*, 2004.

[TP87] Stuart Thompson and Rajnikant Patel. Formulation of joint trajectories for industrial robots using b-splines. *IEEE Transactions on industrial Electronics*, 1987.

[TP98] Jean Taine and Jean-Pierre Petit. *Transferts thermiques, Mécanique des fluides anisothermes*. Dunod, 1998.

[vS98] Oskar von Stryck. Optimal control of multibody systems in minimal coordinates. *Zeitschrift fur Angewandte Mathematik und Mechanik 78, Suppl 3*, 1998.

[Win73] D. Winfield. Function minimization by interpolation in a data table. *Journal of the Institute of Mathematics and its applications*, 1973.

[WW90] Thomas Wonnacott and Ronald Wonnacott. *Statistique : Economie, Gestion, Sciences, Médecine*. Economica, 1990.

[Žla96] L. Žlajpah. On time optimal path control of manipulators with bounded joint velocities and torques. *Proceedings IEEE Int. Conf. on Robotics and Automation*, 1996.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopˆole de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399